REMARKS

In response to the Office Action mailed on June 6th, 2007, Applicant wishes to enter the following remarks for the Examiner's consideration. Claims 13-24 are pending in the application, claim 1-12 are withdrawn.

Specification

The title of the invention has been amended to "Method and Apparatus for Elimination of Prolog and Epilog Instructions in a Vector Processor using Data Validity Tags and Sink Counters."

Rejection of claims under 35 USC §112

Claims 13-24 have been rejected under 35 USC §112 as being indefinite. Applicant respectfully traverses this rejection of the claims in view of the amendment to specification on page 6, lines 1-5. In general, a loop comprises a prolog, a loop body and an epilog. The prolog instructions are placed at the start of the loop, before the loop body and are used for priming the pipeline. The epilog instructions are placed at the end of the loop, after the loop body and are used for draining the pipeline. This use of the terms 'epilog' and 'prolog' is well known to those of ordinary skill in the art. In general a loop comprises the prolog, the loop body (which is repeated) and the epilog. By way of example, the Hennessy and Patterson article from "Computer Architecture: A Quantitative Approach" that is cited by the examiner shows an example loop on page 240. In this loop, the instruction

A[1] = A[1] + B[1];

is at the start of the loop and is the loop prolog. The two instructions

B[i+1] = C[i] + D[i];

A[i+1] = A[i+1] +B[i+1];

form the body of the loop and are repeated 99 times. The instruction

B[101] = C[100] + D[100];

is at the end of the loop and forms the epilog. In contrast, the loop on page 239 has a loop body but no prolog or epilog instructions.

Figures 1 and 2 of the application illustrate the distinction between the prolog, loop body and epilog. Page 8, line 14-17, of the specification explain

that: "At the start and end of the computation, the pipeline is not filled, and the asterisks denote invalid data. In prior processors, the sequences of instructions involving invalid data are coded separately and are referred to as the prolog and the epilog."

In light of the foregoing amendment and remarks, Applicant respectfully submits that the claims are not indefinite and respectfully requests that this basis of rejection of the claims be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

Rejection of claims under 35 USC §102

Claims 13-24 have been rejected under 35 USC §102(b) as being anticipated by Hennessy and Patterson, "Computer Architecture: A Quantitative Approach", 1996, Morgan Kaufman Publishers, Inc., Second Edition, pages 269-247. Applicant respectfully traverses this rejection of the claims.

Claim 13. As described in the specification on page 6, lines 17-22, the prolog is one or more instructions at the start of a loop that are used for priming the pipeline, while the epilog is one or more instructions at the end of a loop that are used for draining the pipeline. The loop on page 240 the instruction

A[1] = A[1] + B[1];

is at the start of the loop and is the loop prolog. The two instructions

B[i+1] = C[i] + D[i];

A[i+1] = A[i+1] +B[i+1];

form the body of the loop and are repeated 99 times. The instruction

B[101] = C[100] + D[100];

is at the end of the loop and forms the epilog. In contrast, claim 13 is a method for executing a pipelined program loop having a loop body but no prolog instructions. One aspect of the present invention is the elimination of prolog instructions. Hennessy has not eliminated the prolog instruction.

A further element of claim 13 is "determining if the input data values are valid by checking the associated input data validity tags." This element is not disclosed by the Hennessy reference. The examiner refers to the scoreboard on page 247 of the Hennessy reference. However, the score

board relates to instruction status not data validity. Each entry in the scoreboard corresponds to an instruction or to a function unit executing an instruction. There are no entries that relate to data validity.

The specification on page 3, lines 7-12 explains that, in certain embodiments consistent with the invention, tags are associated with data elements in the pipeline to indicate the validity of the data. These tags are used to eliminate the prolog code, by suppressing the sinking of data tokens that are not tagged as valid. This approach avoids the use of centralized structures. The epilog instructions are eliminated by using sink-specific iteration counts, which again avoids the use of centralized structures. The scoreboard of Hennessy is an example of a centralized structure. The entries in the score board are not associated with the data. For example, a given instruction may have been completed and the result written, but there is no indication in the scoreboard as to whether the result is valid or not. The result will be invalid, for example, if any of the input operands is invalid. The result may be passed to a subsequent functional unit, which in turn has no way of knowing if the data is valid. The scoreboard allows synchronization between functional units, but cannot be used to eliminate prolog instructions.

In light of the foregoing amendment and remarks, Applicant respectfully submits that the Hennessy reference does not teach, suggest, disclose or otherwise anticipate the recitations of claim 13. Applicant thus respectfully requests that this basis of rejection of the claims be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

Claim 14 includes the recitation "if all of the input data values are valid, performing a functional operation on the input data values, storing the result of the functional operation in the result register and setting the associated output data validity tag to indicate that the intermediate result is valid; and, if any of the input data values is invalid, setting the associated output data validity tag to indicate that the intermediate result is invalid." This element of the claim is not disclosed by the Hennessy reference. The scoreboard of the Hennessy shows if an instruction has been completed and the results are ready, but does not indicate whether the result valid or invalid. For example, the result of an operation is invalid if any of the input operands is invalid. Further, the

scoreboard does not show if the inputs are valid or invalid and so the validity of the output is unknown. Thus, the scoreboard can only be used if all of the data is valid. In turn this means that the pipeline must be primed and that prolog instructions are required.

Claims 15 and 16 depend from claim 13. Although additional arguments could be made for the patentability of each of the claims, such arguments are believed unnecessary in view of the above discussion.

Claim 17 adds the element of a source counter to determine if a data value is valid or invalid. The examiner asserts that the iteration counter 'i' is a source counter, however, applicant respectfully submits that this assertion is incorrect. The iteration counter counts how many times the loop body has been executed. However, source may be accessed multiple times in a single loop while another source is accessed only once. Thus, different sources may have different counter values which, in turn, differ from the loop counter value. Claim 17 further calls for setting the output data validity tag to indicate that the result is invalid if all data values have been retrieved from memory. Elements of Hennessy's scoreboard relate only to functional unit status, they do change dependent upon the value of the iteration counter, i.

Referring to the example loop shown in Figure 4 of the specification, it can be seen that the loop body is executed 5 times. The source counter is decremented at time steps 0, 3 and 6 (each time the source is accessed). The counter has expired at time step 9, so the input value is tagged as invalid. However, two more loop iterations remain to be counted. It is clear from this example that the loop iteration counter is not equivalent to the source counter.

Claim 19, calls for determining the validity of the output data from the associated output data validity tag. This element is not disclosed by Hennessy. As discussed above, the output is invalid if any of the inputs is invalid. Hennessy does not disclose a method for tracking data validity. Merely knowing that an operation is complete is not sufficient to determine its validity when an input may be invalid.

In light of the foregoing amendment and remarks, Applicant respectfully submits that the Hennessy reference does not teach, suggest, disclose or otherwise anticipate the recitations of claim 13-19. Applicant thus respectfully

requests that this basis of rejection of the claims be withdrawn and that a Notice of Allowance for these claims be mailed at the Examiner's earliest convenience.

Claim 20. As described in the specification on page 6, lines 17-22, the epilog is one or more instructions at the end of a loop that are used for draining the pipeline. Referring to the loop on page 240 of the Hennessy reference, the instruction

A[1] = A[1] + B[1];

is at the start of the loop and is the loop prolog. The two instructions

B[i+1] = C[i] + D[i];

A[i+1] = A[i+1] +B[i+1];

form the body of the loop and are repeated 99 times. The instruction

B[101] = C[100] + D[100];

is at the end of the loop and forms the epilog. In contrast, claim 20 is a method for executing a pipelined program loop having a loop body but no epilog instructions. One aspect of the present invention is the elimination of epilog instructions. Hennessy has not eliminated the epilog instruction.

The examiner asserts that the iteration counter 'i' is a sink counter, however, applicant respectfully submits that this assertion is incorrect. Firstly, the examiner has already identified 'i' as a source counter in respect to claim 17. Multiple input values may be used to computer a single output value, so a single counter cannot be used, in general, as both a source counter and a sink counter. The iteration counter counts how many times the loop body has been executed. However, a sink may be accessed multiple times in a single loop. Thus, difference sinks may have different counter values which, in turn, differ from the loop counter value.

Referring to the example loop shown in Figure 3 of the specification, it can be seen that the loop body is executed 5 times. The sink counter is decremented at time steps 6, 9 and 12. The counter has expired at time step 15, so the output value is not sunk. The loop iteration counter is modified in time steps 0-5, while the sink counter is unchanged. It is clear from this example that the loop iteration counter is not equivalent to the sink counter.

Claims 21-24 depend from claim 20. Although additional arguments could be made for the patentability of each of the claims, such arguments are believed unnecessary in view of the above discussion.

In light of the foregoing amendments and explanations, applicant submits that all rejections of claims 13-24 have been overcome. Allowance of claims 13-24 is therefore respectfully requested at the Examiner's earliest convenience. Although additional arguments could be made for the patentability of each of the claims, such arguments are believed unnecessary in view of the above discussion. The undersigned wishes to make it clear that not making such arguments at this time should not be construed as a concession or admission to any statement in the Office Action.

Please contact the undersigned if you have any questions regarding this application.

Respectfully submitted,

/Renee' Michelle Leveque/

Renee' Michelle Leveque

Leveque Intellectual Property Law, P.C.
Reg. No. 36,193
221 East Church Street
Frederick, Maryland 21701
301-668-3073
Attorney for Applicant(s)